Peter the Great St. Petersburg Polytechnic University

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Institute of Computer Science and Technology](http://english.spbstu.ru/structure/institut_computernikh_nauk_i_tekhnologiy/)

Department of Computer Systems & Software Engineering

Lecture

*Nios II Software Tools and Design Flow*

student:

Ivanov Il’ya

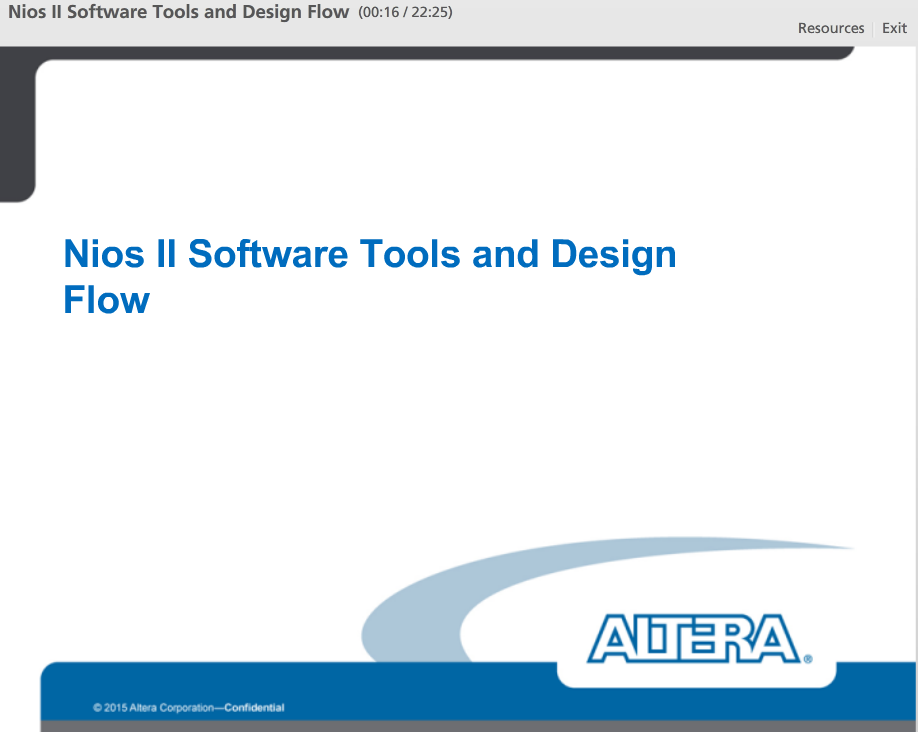
group: 3530901/70203

lecturer:

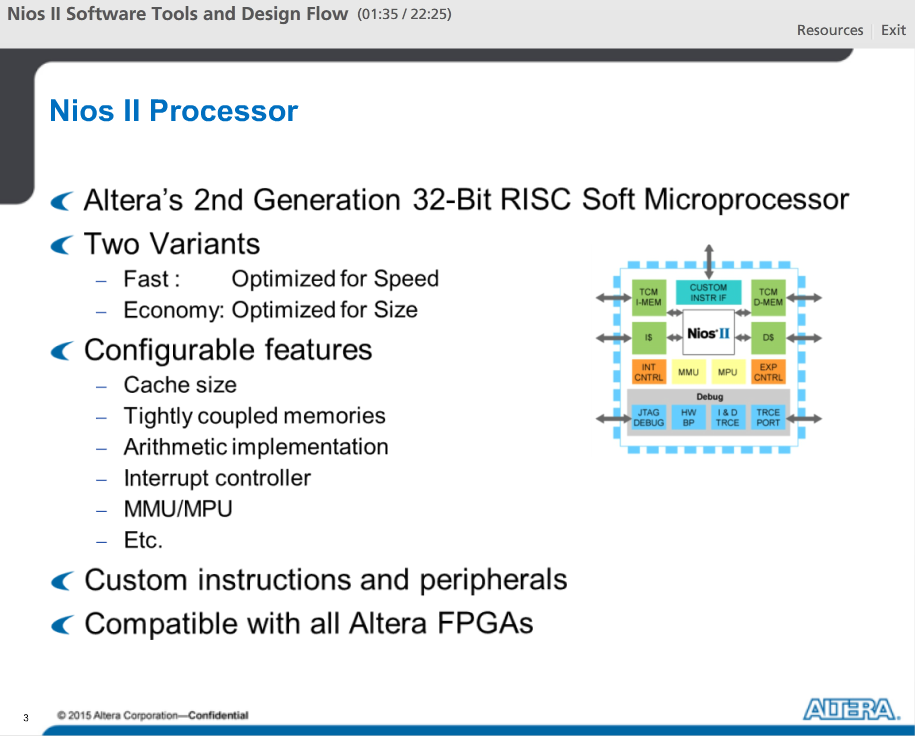
Antonov A.P.

Saint-Petersburg

2020



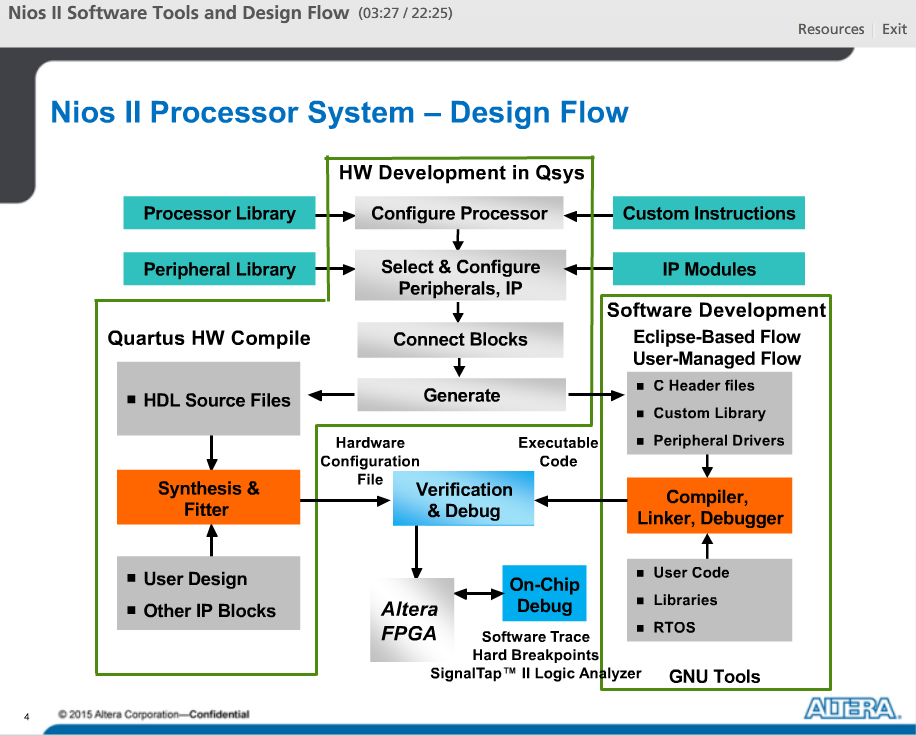
In this lecture, we will give a brief overview of what it takes to successfully create a Nios II processor based system. We will talk about all of the tools available as well as the design flow required.



Nios II Processor is Altera's 2nd Generation 32-bit RISC Soft Microprocessor. It is soft because it can be synthesized on to any Altera FPGA. It is also very versatile coming in two variants: fast optimize for speed and economy optimized for size. It can be used in almost any embedded system. It is a highly capable microcontroller class processor.

The Nios II processor is highly configurable depending on your needs. Some of the configurable options are listed here.

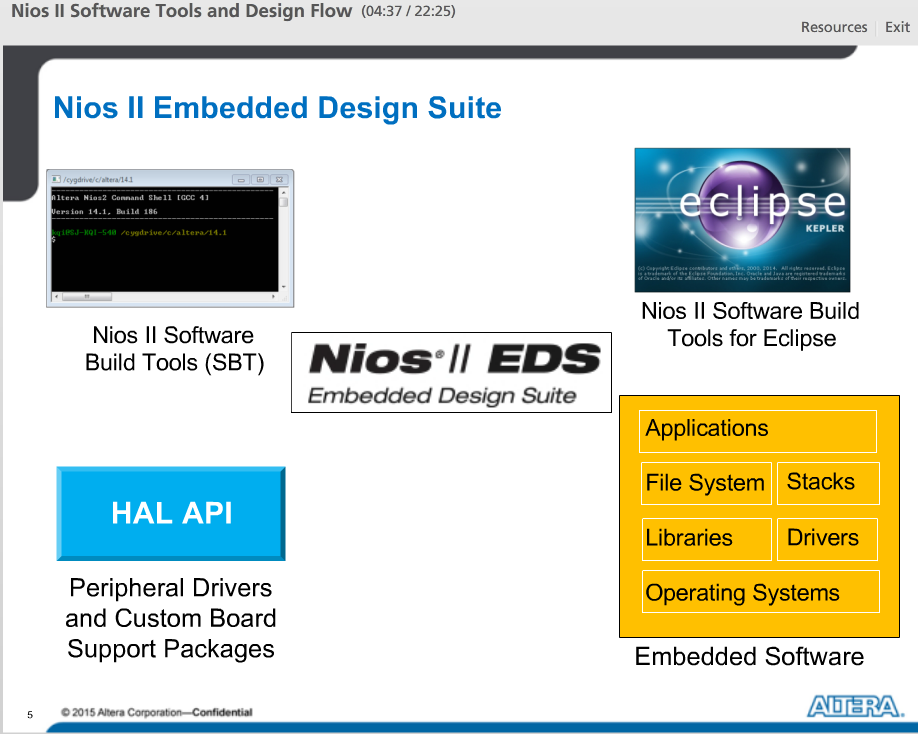
Because the Nios II processor is softcore, you can also implement custom instruction and peripherals to help you accelerate and offload specific tasks.



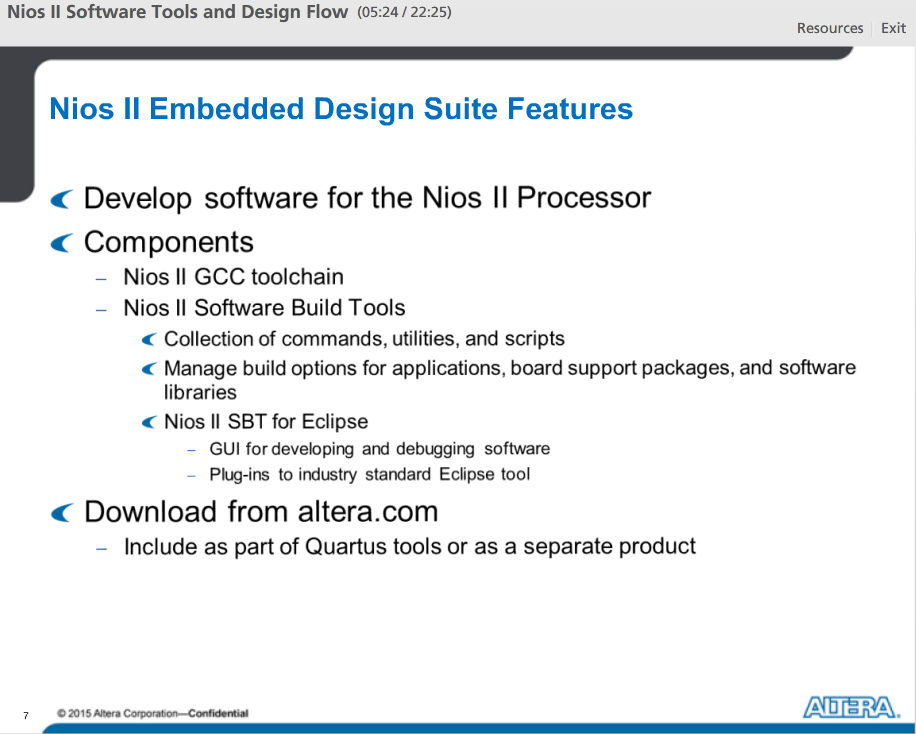
Entire system design flow for a Nios II processor based system.

The process starts at the top with the Qsys system integration tool, which is part of the Quartus Prime software. In Qsys, you would need to configure the options for the Nios II processor, including custom instruction, you can select and instantiate peripheral and IP connected to the processor. After you have instantiate the various blocks, connect them to the Data or Instruction master of the Nios II processor and finally generate the Nios Qsys System. The result of the generate can be used by both the Quartus HW compile flow as well as the software development flow.

On the hardware side, Quartus Prime will compile the generated Qsys HDL and produce a FPGA hardware configuration file that needs to be either downloaded in to the FPGA or burned to flash. The Qsys generate also create handoff files in order for the software build tools to create a custom Board support package. On the software development side, once the projects are created from the handoff files, you can add a lot of custom or off the shelf driver’s libraries and C source files. Add to this OSes or middleware as needed and you will be able to compile your SW project with the given GNU compiler. After an Executable Linkable File is created you would be able to debug this on chip.

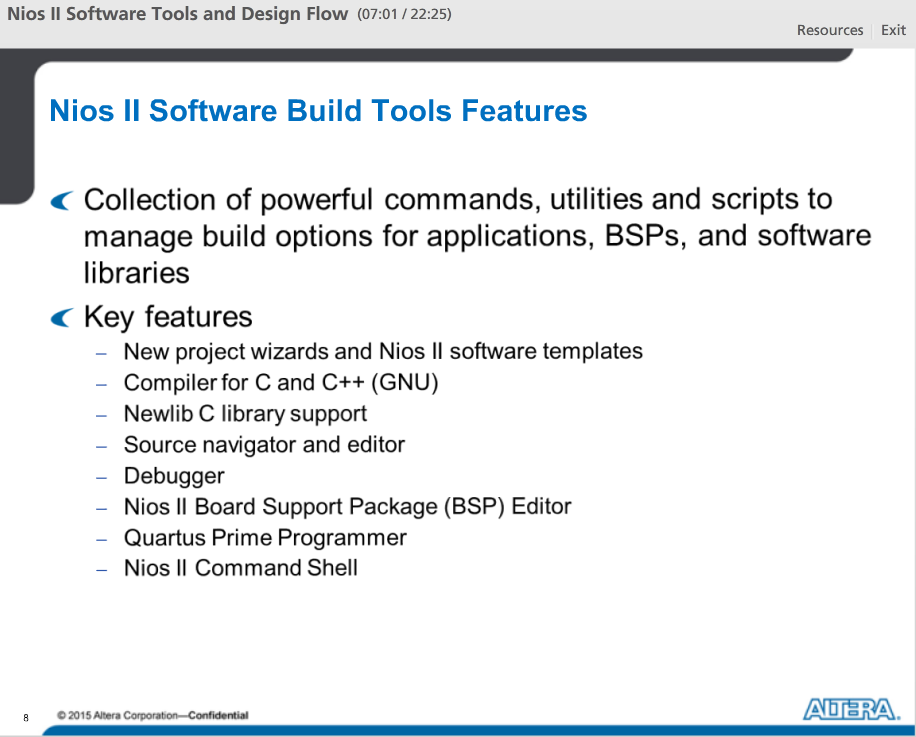


At the center of software development and debugging process is the Nios II embedded design suite. This is a complete tool from Altera, which lets you perform all the usual software development tasks. Within the tool, there is the Software build tools, which allows you to compile, and run you program. The GUI version of the tool is the Nios II SBT for Eclipse. The tool can build automatically a board support package for your system including all the necessary device drivers and component support. The Hardware Abstraction Layer or HAL API allows you to interact easily with components using a Unix-like interface. In your software build tool, you can also add many off the shelf embedded software such as Operating systems, network stacks, file systems, and other software. Once you have finished development, the Software build tool allows you to debug easily over a JTAG connection, which include breakpoint support, run control, and register and memory views.



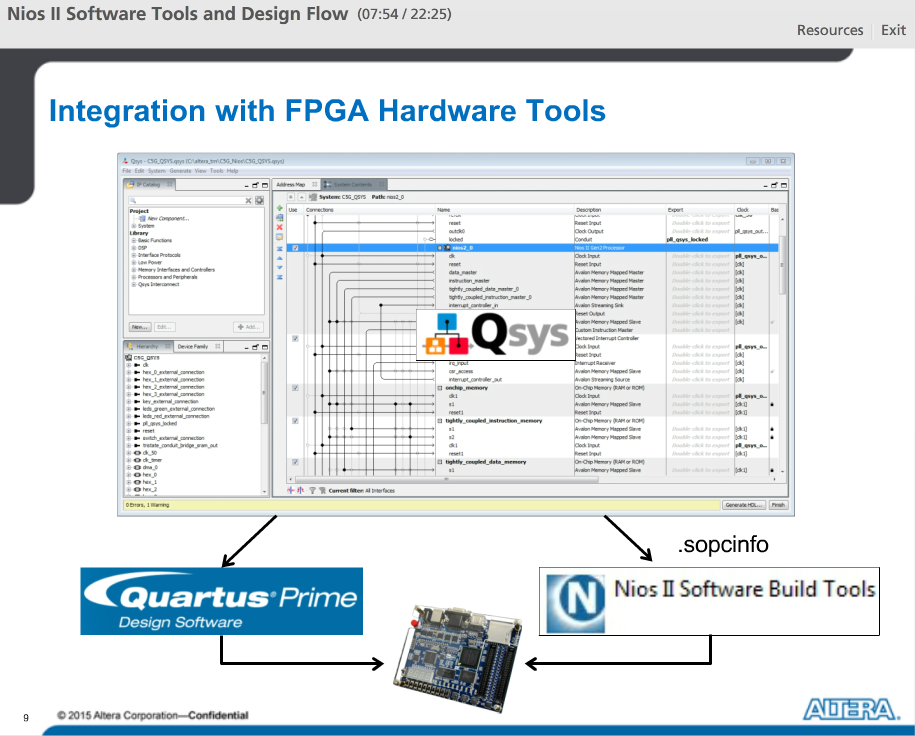
The only tool you need to develop software for the Nios II processor is the Nios II Embedded Design Suite or EDS.

It includes proprietary and open-source tools (such as the GNU C/C++ tool chain) for creating Nios II programs. At the center of the Nios II EDS is the Software Build Tools, which contains commands, utilities, and scripts to help you build applications board support packages and include software libraries. The Nios II SBT for Eclipse gives you plug-ins to industry standard eclipse tool, which makes developing and debugging software easy.



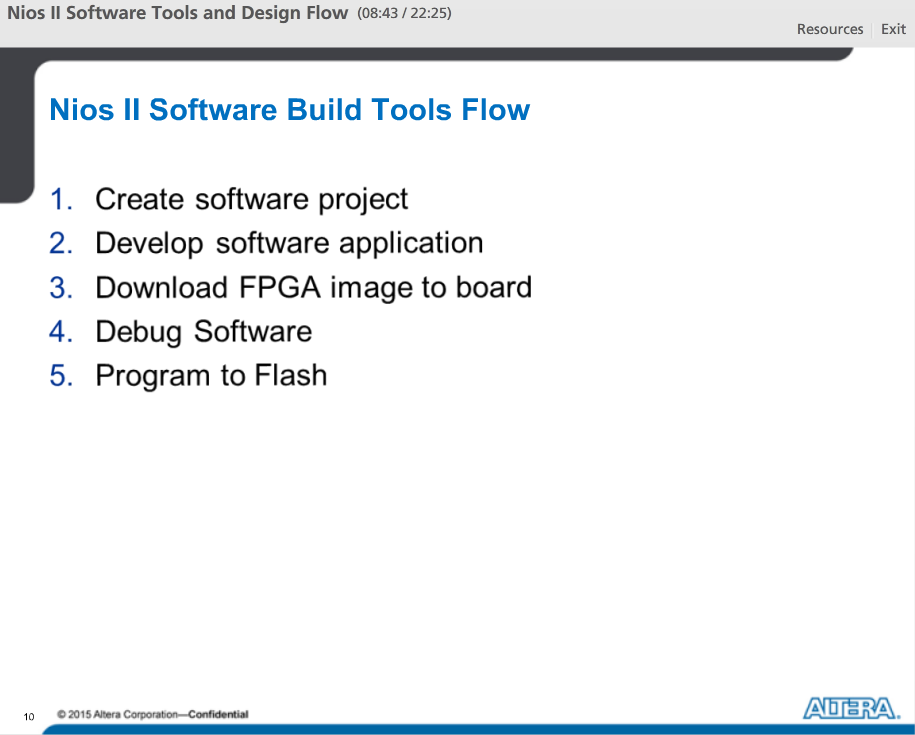
The main component in the Nios II EDS is the Software Build Tools. SBT automate board support package (or BSP) creation for Nios II processor-based systems, eliminating the need to spend time manually creating BSPs. The BSP provides a C/C++ runtime environment from Newlib, compiler from GNU, insulating you from the hardware in your embedded system. Altera BSPs contain the Altera hardware abstraction layer (HAL), an optional RTOS, and the device drivers to talk to the peripherals in your system.

There are two different flows that we can use to take advantage of these software build tools, command line and Eclipse-based. Both allow you to create Nios II software projects, with detailed control over the software build process and both share the same utilities and TCL scripting capabilities.

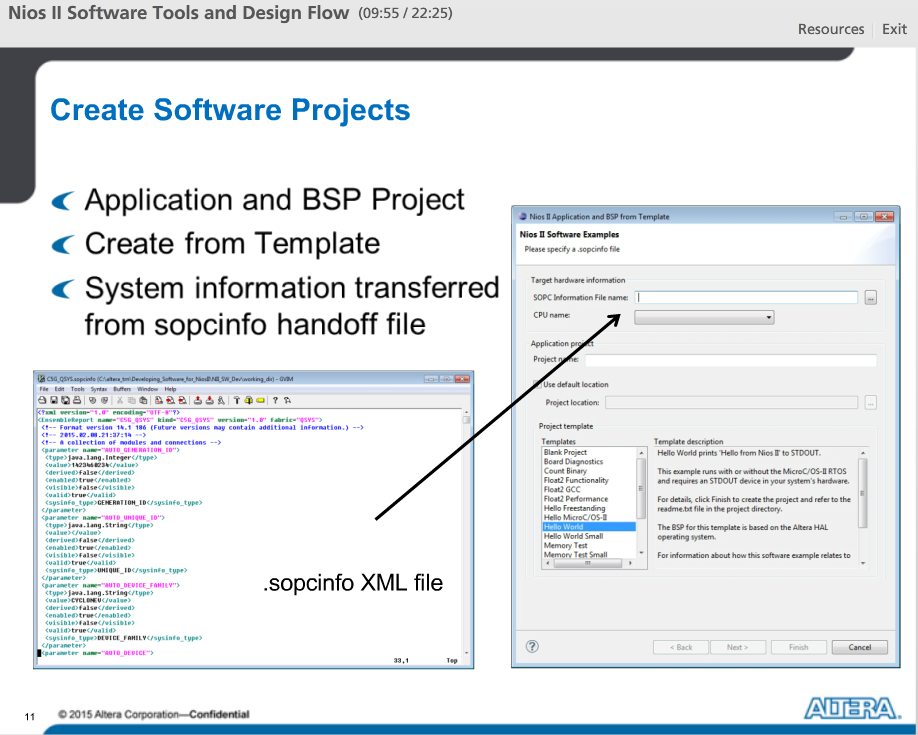


Nios II system hardware is required to be created in the Qsys integration tool, where users can parameterize the Nios II processor and associated peripherals. Once complete, a Qsys generate will create the HDL files to be synthesized by the Quartus Prime Design Software and it will also produce the sopcinfo xml file, which is used to relay important information about the system so the software build tools can automatically build a Board Support Package for the Application Software.

Once the HW compile is done in Quartus Prime SW and the software is compiled with the Software Build Tools, both can be downloaded on the board and you will be able to see the software running on the Nios II processor on the FPGA.



Steps needed to create and debug a software application for the Nios II processor in the Software Build Tools.



The first step is creating a SW project.

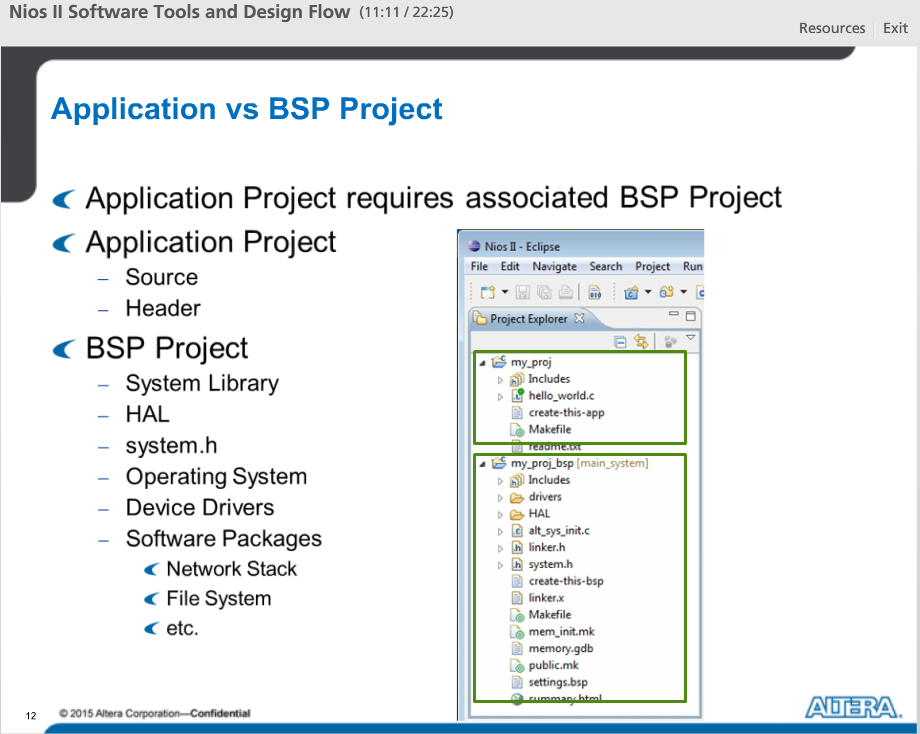
In Eclipse, the easiest way to create a new application and BSP is to use a template. The project template dialog will ask you for the location of your sopcinfo file which is created when you generate your Qsys system.

This sopcinfo file is an XML based file that describes the Qsys system and is used to configure the software.

The tool uses the information about your HW system to create a system library that targets your specific HW, known as the BSP.

You can customize the BSP with optimization settings, different software packages, or specify which peripherals will be used for stdio or stderr.

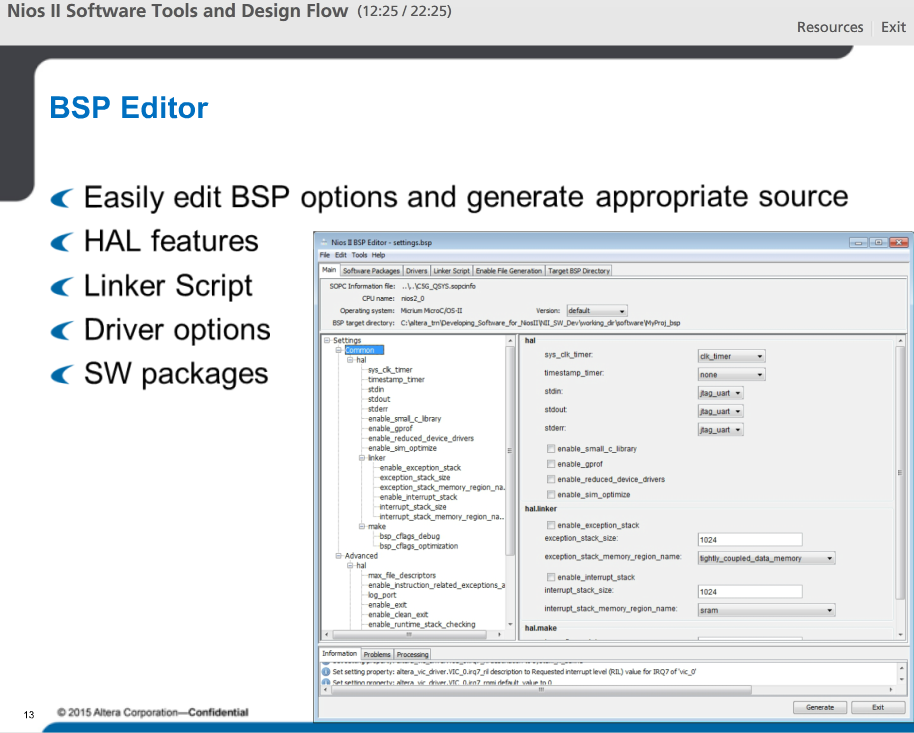
Once you launched your first build, SBT will create the BSP system library for you. It will contain all of the options you have requested and it will generate the system header file that will contain all of the macro definitions for the system parameters as #defines.



When you create the project there will actually be two projects that you can see in the Project explorer pane of the Eclipse software. These are the software application and the BSP.

A Nios II C/C++ application project consists of a collection of source and header files, along with an associated makefile. A typical characteristic of an application is that one of the source files contains function main(). An application can includes code that calls functions in other SW libraries and BSPs. The makefile compiles the source code and links it with a BSP and one or more optional libraries, to create an elf file.

A Nios II BSP project is a specialized library containing system-specific support code. This includes device drivers, OS files, header files, HAL, and any packages you may have added to the system such as a network stack or file system. The BSP project source is automatically created by the Software Bulld Tools.



To make the creating and the customization of the BSP project easier, the Nios II SW Build Tools include the BSP Editor, which can automatically generate various source files for the Nios II BSP project.

In this tool, you will be able to easily edit, save, and generate sources for the BSP.

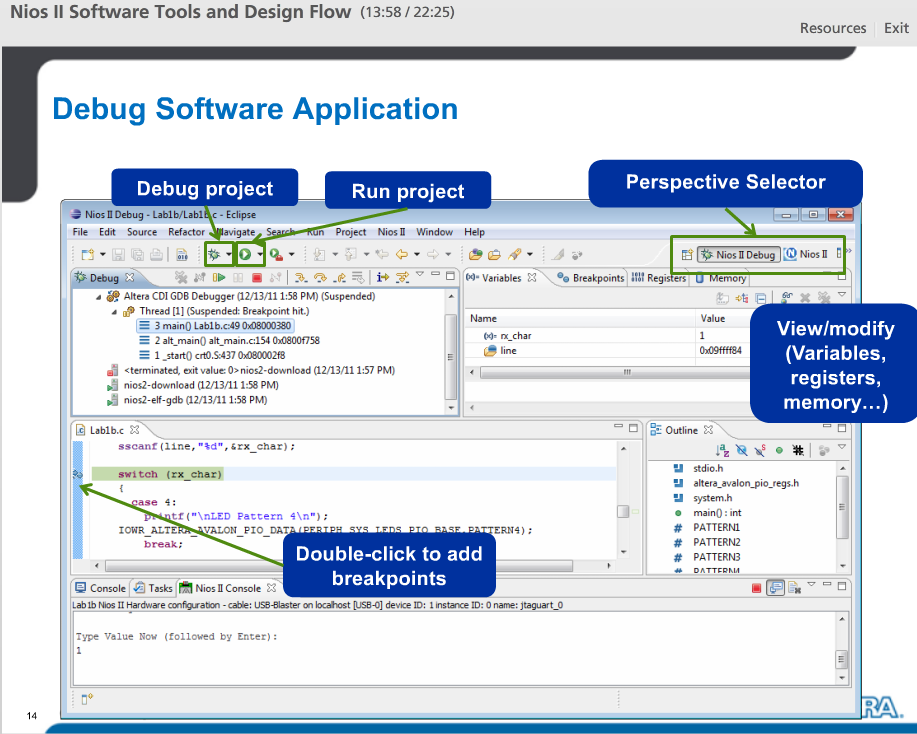
Within the BSP editor, you will be able to select HAL feature options such as standard in/out components, timer components, enabling profiling and others.

You will also be able to create and set memory regions and assign to specific software sections, which is used to generate the linker script.

In the BSP editor, you will have the option to choose driver settings if those are available.

Lastly, many software packages such as interniche's niche stack can be enabled in the BSP Editor.

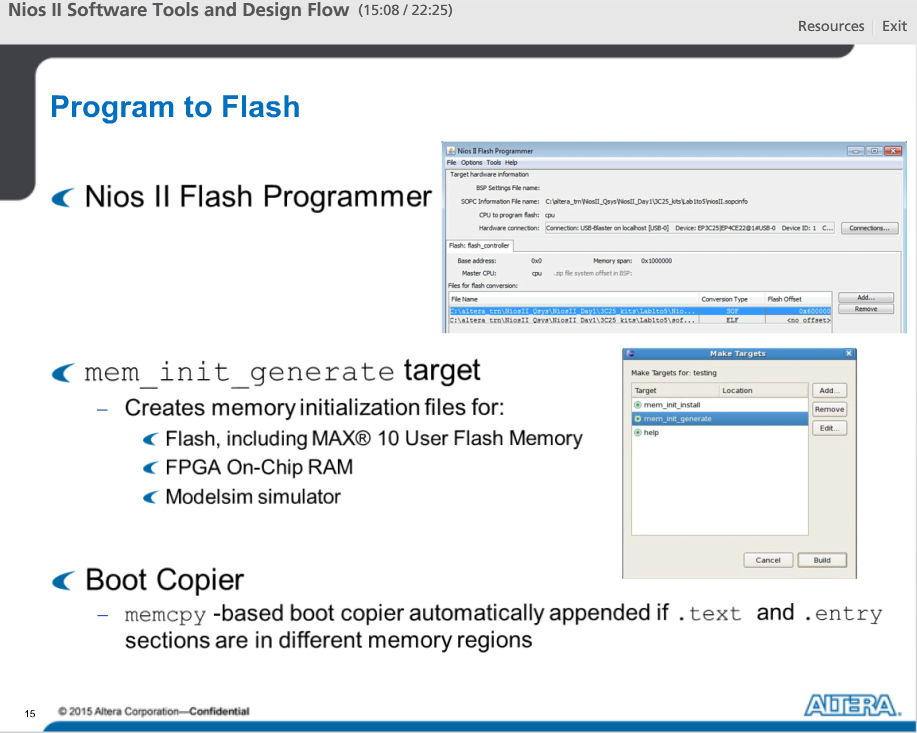
Once selections are made all you need to do is hit the generate button and the BSP project will automatically be updated with new relevant source files.



The Nios II EDS for Eclipse can be used easily and effectively to debug Nios II applications across JTAG.

When you launch the application, the tool will automatically switch over to the Nios II Debug perspective, which include many windows, such as breakpoints and debug run control to make debugging easier.

The debugger graphic interface contains all of the usual features that you would expect from a modern debug tool.

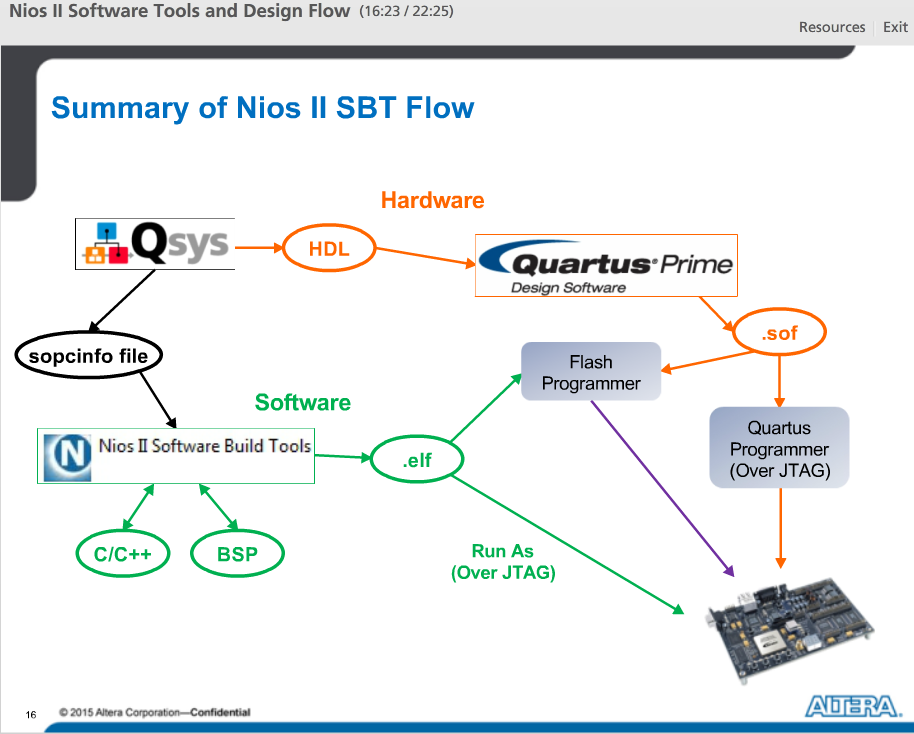


Once you have finished debugging the software you will need to flash the board. There are many ways to do this depending on your device.

Nios II Flash Programmer can be used with any Flash component connected to the Nios II Processor.

For certain scenarios, you can also use the mem\_init\_generate makefile target. With this target, enabled in the SBT, you will be able to create memory-initializing files for Max10 User Flash Memory, FPGA on chip ram, if you plan to boot from those memories. This target will also generate a data file for ModelSim simulator if you choose to simulate your Nios II system there.

Lastly, when generating the flash image using either the flash programmer or mem\_init\_generate target, if the tool detects that you will be running the software in a different memory component from where it's stored (basically, the .text and .entry sections are in different memory regions). Then the Altera tool will automatically append a memcpy based boot copier at the front of your software image.



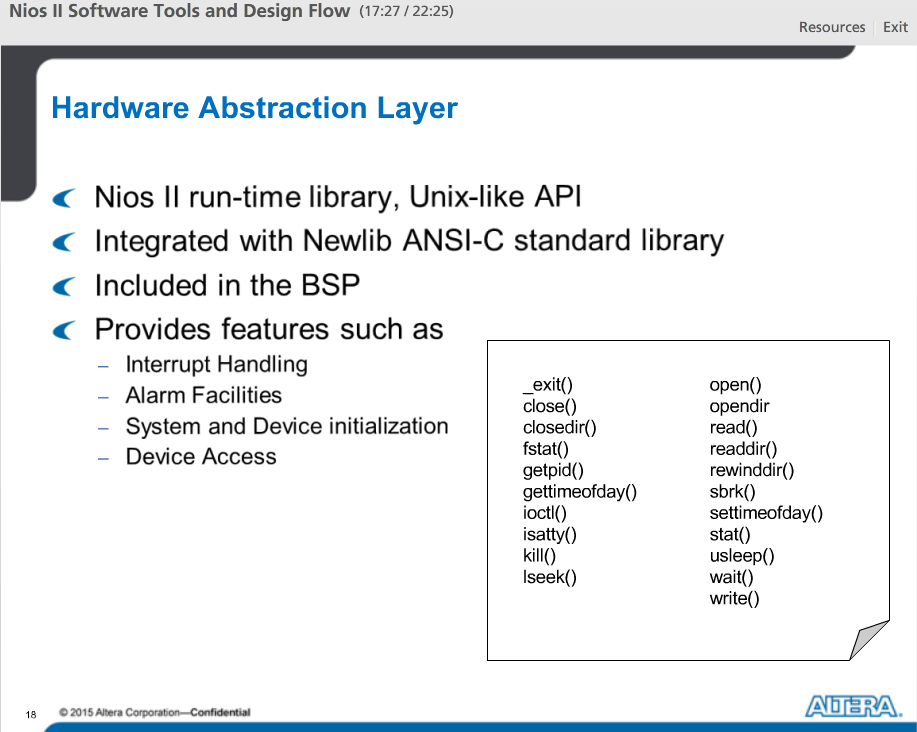
We start with the Qsys system, where hardware configuration is made.

Once you generate the Qsys system, the sopcinfo file, containing hardware information, is created and can be used to generate your BSP project when you create your project. SBT for Eclipse will use the information stored in the sopcinfo file to create a system library or BSP that is used to support all of the custom hardware functionality that you configured into your system with Qsys. Using SBT, you can then add your application files and compile the system executable or ELF file.

The HDL, generated by Qsys, is used by the Quartus Prime software to generate the FPGA configuration file or .sof file. This is then used by the Quartus programmer to configure the FPGA device and download your Nios II system into the hardware.

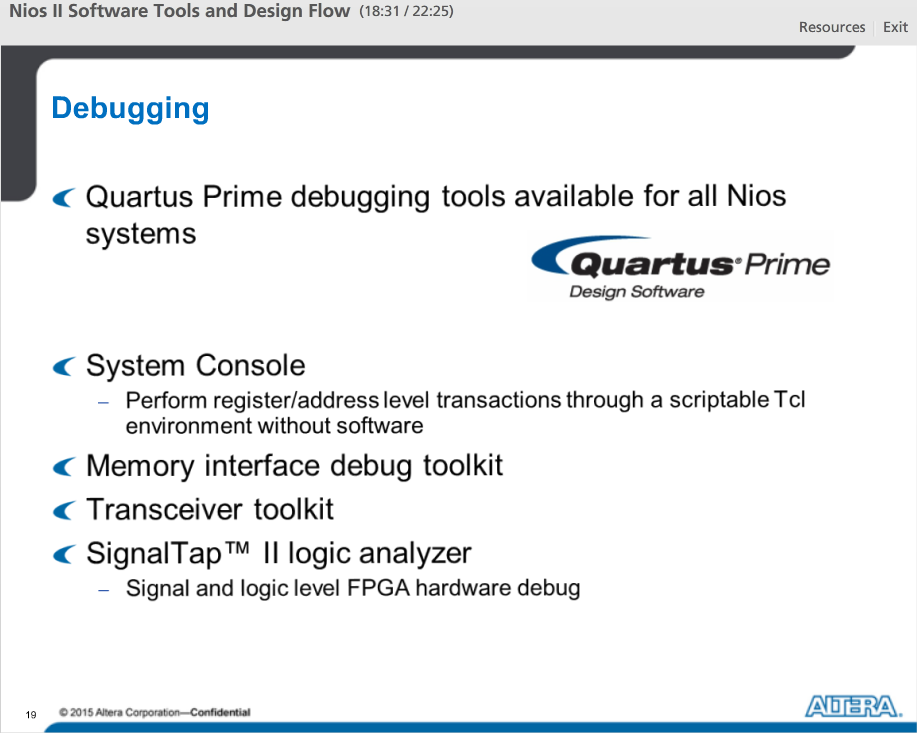
Once the HW is configured, you can use the debugger to download, run and debug the application on the hardware.

Lastly, you can use the software executable together with the FPGA configuration file to build a flash image, which can then be downloaded into the flash on your board. This will enable your system to boot and run standalone.



The HAL offers a UNIX style API for interfacing to peripherals and systems services like timers and interrupts - all hardware settings are abstracted using #define statements that have been automatically generated by the Software Build Tools environment. The Tools will also automatically update all hardware related software libraries.

The HAL provides interrupt handling calls, timer and alarm facilities, device access calls, as well as system and device initialization calls. There are also ways to register custom functions on custom components so you can utilize C or UNIX calls to access your component.



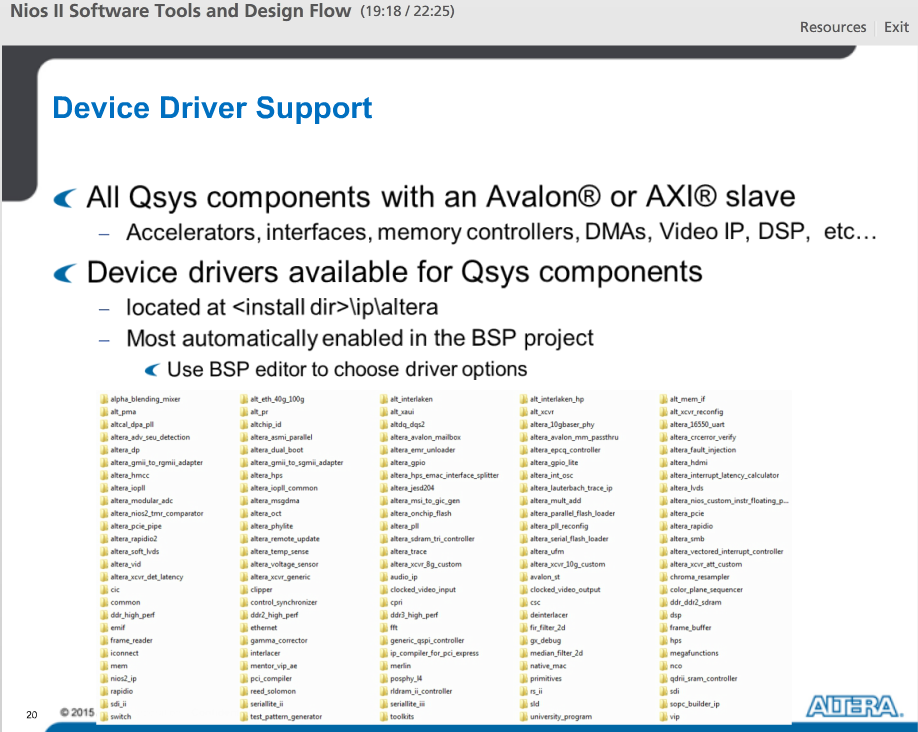
With Nios II systems, all Quartus Prime Debugging tools are also available for use.

System Console is a great tool that lets you perform register and address level transactions through a scriptable environment so you can see the state of your system and perform reads and writes to peripherals in your system without software.

The EMIF Toolkit lets you diagnose and debug calibration problems and produce margining reports for your external memory interface.

Transceiver Toolkit uses System Console technology to help FPGA and board designers validate transceiver link signal integrity in real time in a system and improve board bring-up time.

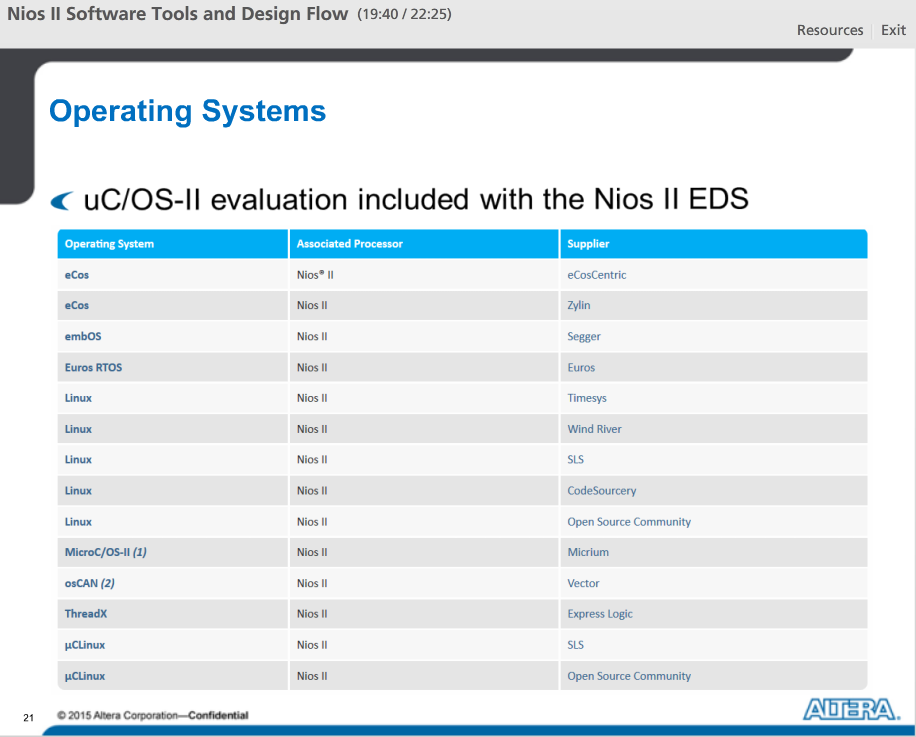
Lastly, for times where you need to see the signal and logic levels of internal FPGA logic, the SignalTapII Logic Analyzer is available for that purposes.



In terms of peripherals, all Qsys components with an Avalon or AXI slave port can be used with the Nios II Processor.

For most off the shelf Qsys components, device drivers are also available. For Altera components they are located at the Quartus install directory\ip\altera directory.

Most of these are automatically enabled in your Nios II BSP project when the component is included in your Qsys system. You can however use the BSP editor to select driver options if they are available.



List of current Operating System support for Nios II Processor.

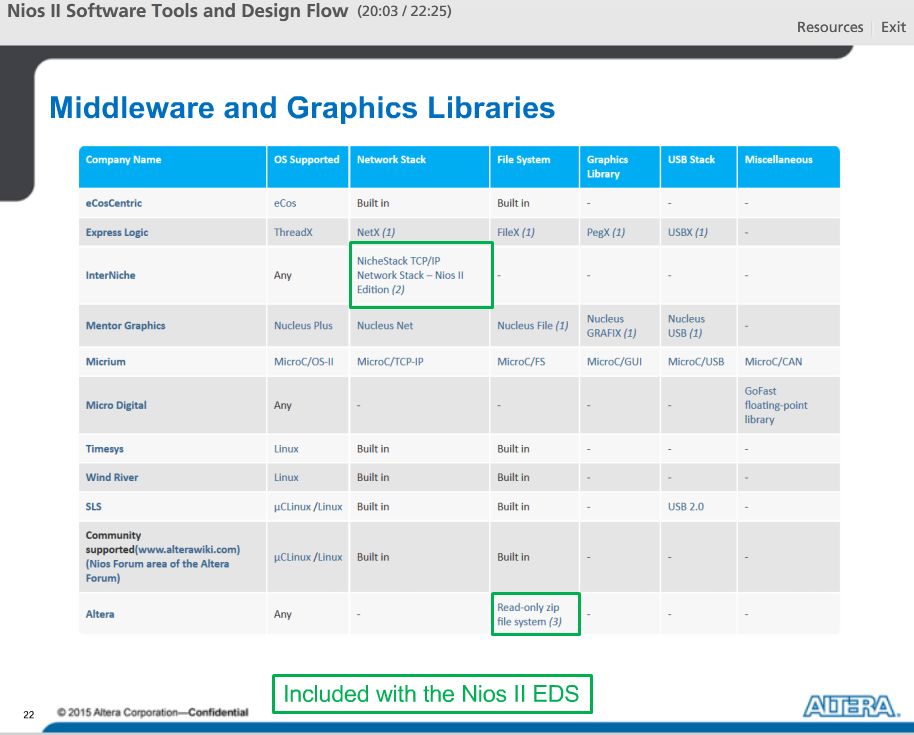


Chart of middleware and graphics libraries available from various vendors.

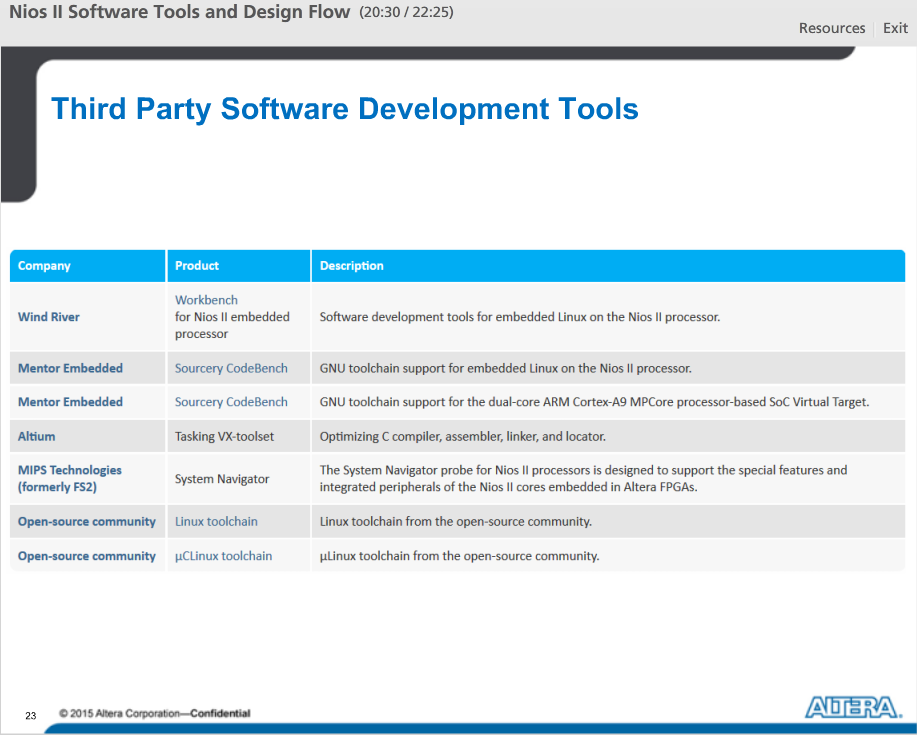


Table of additional third party software development tools available for the Nios II processor in case the default Altera's SBT does not meet your needs.